DC11149
May 2020
R. Ertolupi

# Significant Upgrade for Serial NOR Flash Memories

## 1. Introduction

This document aims to explain the changes, the improvements, and the new features of our new drivers for SPI NOR memories: SerMem4 and SerMem8.
As you may know, these devices are extremely common and they are often the ones which determine the flashing time of a panel. So, being able to optimize operations for them, means that we can reduce the flashing time for most of our customers' applications.

The reason why we started this activity was initially just to support those new memories with the Octo-SPI interface and, since Quad-SPI is a sub-set of Octo-SPI, we decided to extend the new features and improvements also to all the SPI memories.
During this development, we worked together with silicon producers to take care of the small details about these devices. Even if it's true that most of these devices are meant to be compatible and exchangeable, but going deep into the configurations and features they are not compatible at all and we had to manage those peculiarities to exploit the full potential of any device.

We went so deep that now these new drivers are supporting any Quad-SPI and Octo-SPI memory from the following silicon producers:

- Adesto (Atmel)
- Cypress (Spansion)
- ISSI
- Macronix
- Micron
- Winbond

We also joined the Xccela consortium, an association founded by Micron which promotes the Xccela bus as an open standard.

## 2. Contents

## 3. How do serial NOR flash memories work?

In this section, we want to explain what operations are performed in the target devices and how these operations work. This knowledge may help you to optimize even more your application. These are the standard commands:

- **Masserase**

  This command erases all the content of the serial NOR flash memory (all bits are set to 1). Optionally, it's also possible to send two additional parameters to this command: the address from where to start erasing and the number of bytes to erase. Then the driver will automatically manage the sectors to do that operation as fast as possible according to user's requests. Usually, 4KB is the minimum sector size allowed for this operation.

  The duration of this operation depends only on the target device characteristics and there is not much that we can do to improve this. For some devices it could be extremely long, even longer than all the other operations together. For very few devices, this operation may be shorter according to the amount of data programmed in the memory.
  The suggestion is to skip this operation in case the device is virgin. This can be done using a conditional script based on the result of the blankcheck operation:

  ```
  #IFERR TPCMD BLANKCHECK F
  #THEN TPCMD MASSERASE F
  #THEN TPCMD BLANKCHECK F
  ```

- **BlankCheck**

  This command reads all the content of the serial NOR flash memory and checks that all bits are set to 1. This operation is terminated instantly when a 0 is found.
  Optionally, it is also possible to send two additional parameters to this command: the address from where to start reading and the number of bytes to check.

  The duration of this operation depends only on bitrate that has been set by the user. The bit rate directly depends on the protocol and by the frequency, so the suggestion is to use the highest frequency and the fastest protocol. See the examples below which show the time estimation for the blankcheck operation over a 512Mbit serial NOR flash memory:

| Protocol | Frequency | Bitrate | Time for 512Mbit |
|---|---|---|---|
| SPI | 1 MHz | 1 Mbps | 512 s |
| Quad-SPI SDR | 25 MHz | 100 Mbps | 5.12 s |
| Octo-SPI SDR | 50 MHz | 400 Mbps | 1.28 s |
| Octo-SPI DDR | 30 MHz | 600 Mbps | 0.86 s |

- **Program**

  This command takes the customer's data from the FRB file and programs them into the memory.
  Optionally, it is also possible to send two additional parameters to this command: the address from where to start programming and the number of bytes to be programmed.

  The duration of this command mainly depends on the target device but also on the bitrate. In fact, the FlashRunner sends the data of a single page (typically 256 bytes) to the device and then it must wait for the page to be programmed. So, having a higher bitrate only reduces the "send page" time, which usually means saving just 20% or 30% of the command time or even less depending on the target device timings.
  The suggestion to improve the performances of this command is to set the FRB file with the "IGNORE_BLANK_PAGE" option. This will skip the program operation for any page which contains only 0xFF bytes.

  ```
  #TPSETSRC myData.frb IGNORE_BLANK_PAGE
  ```

- **Verify**

  This command reads all the content of the serial NOR flash memory and checks that it corresponds to FRB data. This operation is terminated instantly when a mismatch is found.
  Optionally, it is also possible to send two additional parameters to this command: the address from where to start reading and the number of bytes to check.

  The duration of this operation should be the same as blankcheck, but with a very small delay because we need to manage and check data. When using the new drivers with an OS > 3.00, this delay is almost irrelevant.
  To improve the duration of this command, follow the same suggestions proposed for blankcheck and program commands.

After this explanation should be clear what can be improved by the user, what can be improved by SMH, and what cannot be improved because it depends only on the characteristics of the target device.

## 4. Protocols and frequencies supported

With the new drivers we ampliated the list of supported protocols: we improved the support for Quad-SPI, added the support for Octo-SPI, and we are also giving the possibility to use DDR for some memories. Let us see in detail what this means and how it can be used.

First of all, we must know that the protocol defines that messages are composed of three elements: command, address, and data. These three elements can be transmitted differently, these are all the configurations supported by our new drivers:

- Command, address, and data using a single line (**1-1-1**). This is the standard SPI protocol.
- Command and address using a single line and data using four lines (**1-1-4**). This is called Extended-SPI, Quad-SPI, or just Quad.
- Command, address, and data using four lines (**4-4-4**). This is typically called QPI.
- Command and address using a single line and data using eight lines (**1-1-8**). This is called Extended-SPI, Octo-SPI, or Octal.
- Command, address, and data using eight lines (**8-8-8**). This is typically called OPI.

The names used for the configurations above are not always common between the various silicon producers, that is why we preferred to simplify the choice to the user between these three options:

1. **SPI**
   This is just the standard SPI (**1-1-1**).

2. **Quad-SPI**
   This includes both options which use four lines for data (**1-1-4**) and (**4-4-4**). Since there are no relevant performance differences between them, we hid this choice to the user and to let the driver decides which is the best option to use according to the device characteristics (not all devices offer both options).

3. **Octo-SPI**
   This includes both options which use eight lines for data (**1-1-8**) and (**8-8-8**). About this design choice, the considerations are the same as Quad-SPI.

Moreover, we added the support for **DDR** (Double Data Rate), also said DTR (Double Transfer Rate), which allows us to double the bitrate without doubling the frequency of the protocol clock. In fact, when using SDR (Single Data Rate), signals are sampled only on the rising edge of the clock, while, for DDR, signals are sampled on both rising and falling edges. This feature is not available for all devices.

In case the chosen device does not support DDR and the hardware setup does not support Quad-SPI or Octo-SPI, there is the possibility to increase the frequency and this has also been improved. When using the new drivers with an OS > 3.00, also these frequencies higher than 25.0 MHz can be selected: 27.27MHz, 30.00MHz, 33.33MHz, and 37.50MHz.

If you notice some instabilities when rising frequency or when trying to use Quad-SPI or Octo-SPI, it could be related to the hardware setup (see the next chapter).

## 5. Hardware setup

The **standard SPI** protocol specifies four fundamental signals: *Chip Select*, *Clock*, *MOSI* (Master Output Slave Input), and *MISO* (Master Input Slave Output). Serial NOR flash memories also have two more signals which need to be managed: *WP* (Write Protect) and *Hold*. These two lines must be kept high while operating on the target device using SPI protocol and, in case the customer's board does not allow them to be set high, then there may be a design problem.

The **Quad-SPI** used by serial NOR flash memories uses the same connections of standard SPI, it just changes the function of these four lines which are used as data lines:

- *MOSI* becomes *IO0*
- *MISO* becomes *IO1*
- *WP* becomes *IO2*
- *Hold* becomes *IO3*

So, there is no needs to change wirings to use Quad-SPI instead of SPI.

The **Octo-SPI** protocol is similar to Quad-SPI, just with four additional data lines: *IO4*, *IO5*, *IO6*, and *IO7*. This means that the backward compatibility with SPI is still maintained.

As explained in the previous chapter, using Quad-SPI or Octo-SPI gives a big advantage because the bitrate can be much higher, but there is also a disadvantage: a good hardware setup is required. In fact, while SPI is very robust and can work also in bad conditions, Quad-SPI and Octo-SPI are more sensitive because they have many synchronous data lines and the quality of the wirings is crucial:

- Use our cable interface to go as near as possible to the target device.
- The wiring should be as short as possible.
- All the wiring should have all the same length.
- There should be as few discontinuities as possible (i.e. prefer one 30 cm cable instead of two 15 cm cables connected together).

Another common issue that can arise when multiple lines move at the same time is the crosstalk: the phenomenon by which signals transmitted on one or more lines generate undesired effects on other lines due to electromagnetic coupling between them. For example, when transmitting 0x00 followed by 0xFF in Octo-SPI, the simultaneous switching of eight data lines from 0 to 1 can pull the clock line to 1 and generate a spurious clock pulse.
To mitigate this effect, one possible solution is to add small resistors (i.e. 100 Ohm) in series on all data lines. These resistors decrease the current flow on data lines and then reduce the interaction effect on the clock line.

## 6. Register operations

Compared to the older version of SerMem, in the new drivers SerMem4 and SerMem8 we extended the support for read/write register operations. We added these three commands: **`WRITE_REG`**, **`VERIFY_REG`**, and **`READ_REG`**. These commands are very flexible and they change their functionality according to the target device characteristics.

For SerMem4, these are the command descriptions:

- **`WRITE_REG <registerName> <value>`**
  Write register command: write the value into the selected register.
- **`VERIFY_REG <registerName> <value>`**
  Verify register command: verify that the value of the selected register corresponds to the one requested.
- **`READ_REG <registerName>`**
  Read register command: read the value of the selected register.

The **`registerName`** parameter must be chosen from the tables below according to the family of the target device:

| Device Family | Register Name | Description |
|---|---|---|
| **AT25Q** and **W25** | SR | Status Register |
| | SR2 | Status register 2 |
| | SR3 | Status register 3 |
| **IS25_P** | SR | Status Register |
| | NVRR | Non-volatile read register |
| | VRR | Volatile read register |
| **IS25_Q** | SR | Status Register |
| **MT25Q** | SR | Status Register |
| | NVCR | Non-volatile configuration register (2-byte) |
| | VCR | Volatile configuration register |
| **MX25**, **MX66**, and **S25FL-S** | SR | Status Register |
| | CR | Configuration register |
| **S25FL-L** and **S25FS-S** | SR | Status Register |
| | CR1NV | Non-volatile configuration register 1 |
| | CR2NV | Non-volatile configuration register 2 |
| | CR3NV | Non-volatile configuration register 3 |
| | CR1V | Volatile configuration register 1 |
| | CR2V | Volatile configuration register 2 |
| | CR3V | Volatile configuration register 3 |

Please, see the wiki on our website to get the updated list of the supported operations. Don't hesitate to ask for support in case of doubts because operating on registers could be critical.

For SerMem8, the commands are a bit different because devices have different characteristics:

- **`WRITE_REG <registerName> <address> <value>`**
  Write register command: write the value into the selected register. The address parameter is needed only for some particular registers.
- **`VERIFY_REG <registerName> <address> <value>`**
  Verify register command: verify that the value of the selected register corresponds to the one requested. The address parameter is needed only for some particular registers.
- **`READ_REG <registerName> <address>`**
  Read register command: read the value of the selected register. The address parameter is needed only for some particular registers.

The **`registerName`** parameter must be chosen from the tables below according to the family of the target device:

| Device Family | Register Name | Description |
|---|---|---|
| **ATXP** | SR | Status register |
| | SR2 | Status register 2 |
| | SR3 | Status register 3 |
| | IOCR | I/O Pin Drive Strength Control Register |
| **IS25_X** and **MT35X** | SR | Status register |
| | NVCR [Addr] | Non-volatile configuration registers (address required) |
| | VCR [Addr] | Volatile configuration registers (address required) |
| **MX25_W**, **MX25_M**, **MX66_W**, and **MX66_M** | SR | Status register |
| | CR | Configuration register |
| | NVCR2 [Addr] | Non-volatile configuration registers 2 (address required) |
| | VCR2 [Addr] | Volatile configuration registers 2 (address required) |
| **S28HS_T** and **S28HL_T** | SR | Status register |
| | AR | Any register (address required) |

Note: the address parameter must be taken from the documentation of the target device.

Please, see the wiki on our website to get the updated list of the supported operations. Don't hesitate to ask for support in case of doubts because operating on registers could be critical.

## 7. Verify using CRC

As you may know, CRC is a powerful tool widespread in many applications nowadays, mostly to quickly compare big amounts of data and detect changes.
CRC is the acronyms of “Cyclic Redundancy Check” and it is an algorithm that calculates a code based on the data and their position. It is specifically designed to protect against accidental communication error, where it can provide quick and reasonable assurance of the integrity of data.
However, this algorithm has some limits because it uses a surjective function, associating to a single CRC value more combinations of data. This means that it is not suitable for protecting against intentional alteration of data, we underlined “intentional” because it is extremely difficult and highly improbable to get the same CRC from similar data.

We added this feature for those devices which have a built-in CRC calculator, such as the Micron’s devices (MT25Q and MT35X) and some Cypress’ devices (S28HS-T and S28HL-T).
Cypress implements a CRC-32 function which means it is using a 32-bit value to express the CRC value (4,294,967,296 different possible values). While, Micron implements a CRC-64 function which means it is using a 64-bit value to express the CRC value (9,223,372,036,854,775,808 different possible values). It should be clear that having more bits corresponds to have higher assurance.

With our new drivers, the customer can choose to use these features to verify that the CRC of the memory content corresponds to the one of the expected data. This verify method is typically faster than reading out all the data from the memory, it could result slower just if the amount of data is not too big or if the selected bitrate is very high.
The user can choose between one of the three methods below to invoke the verify CRC command:

- **VERIFY F S**
  This command calculates the CRC of the FRB file and compares it with the one calculated by the device. Usage example:

```
#TPCMD VERIFY F S
```

- **VERIFY F S <expectedCRC>**
  This command takes the pre-calculated CRC value and compares it with the one calculated by the device. This may result faster because the FlashRunner does not need to spend time calculating the CRC of the FRB file. Usage example:

```
#TPCMD VERIFY F S 0xEE2D496C36202742
```

- **VERIFY F S <startAddress> <size> <expectedCRC>**
  This command is equivalent to the previous one, but it can be applied to a portion of the memory. Usage example:

```
#TPCMD VERIFY F S 0x00000000 0x04000000 0x3E03A198
```

To get the value to use as expected CRC parameter, the customer can use one of the following commands:

- **CALC_FRB_CRC32**
- **CALC_FRB_CRC64**

These commands can be executed by FlashRunner without being connected to the target device because it is just an internal calculation and they will return the commands to use in the real-time log. Usage example:

```
...
#TPSETSRC myFirmware.frb
#TPSTART
#TPCMD CALC_FRB_CRC64
#TPEND
```

We also decided to use CRC to perform the blankcheck for these devices because it resulted faster in most of cases. This is completely hidden to the user because calling the standard command for the blankcheck operation, it will be automatically redirected to a CRC check.

If the customer still prefers to use the classic blankcheck method by reading out all data, it is possible to use blankcheck commands with the address and size parameters and selecting the entire memory space. Example to use the classic blankcheck instead of using CRC for a 512Mbit memory:

```
#TPCMD BLANKCHECK F 0x00000000 0x04000000
```

## 8. Multiple memories on the same bus

Sometimes it happens to have more than a single memory on the same SPI bus different CS for anyone of them. This could happen because it is the device which is actually containing two memories, for example S70FL01GS is containing two S25FL512S, but this could be also a design choice of the customer.

In any case, we improved the flexibility of our new drivers to be able to manage up to three chips on the same bus with a single ISP channel. Before this optimization, it was necessary to connect a single ISP channel to each one of the serial NOR flash memories and to operate on them sequentially while, now, it is possible to manage them in parallel and this means saving a lot of time.

At the moment, in our list of the supported devices there are just two devices that use this special feature: the S70FL01GS, as mentioned before, and the MT35XU512_x3, which is a customization done for a customer who has three MT35XU512 on the same bus.
Using this new feature, we could improve the flashing time of about 70% for both of them.

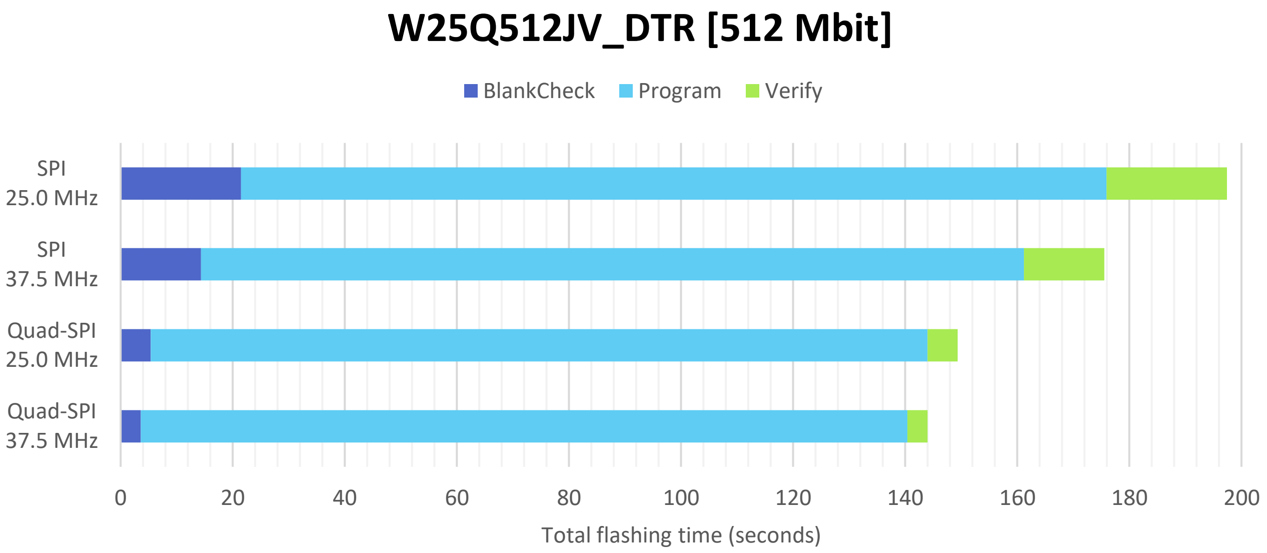
This feature is available only for those devices which require it by default (like S70FL01GS) or if it is explicitly requested by the customer.

## 9. Flashing time examples

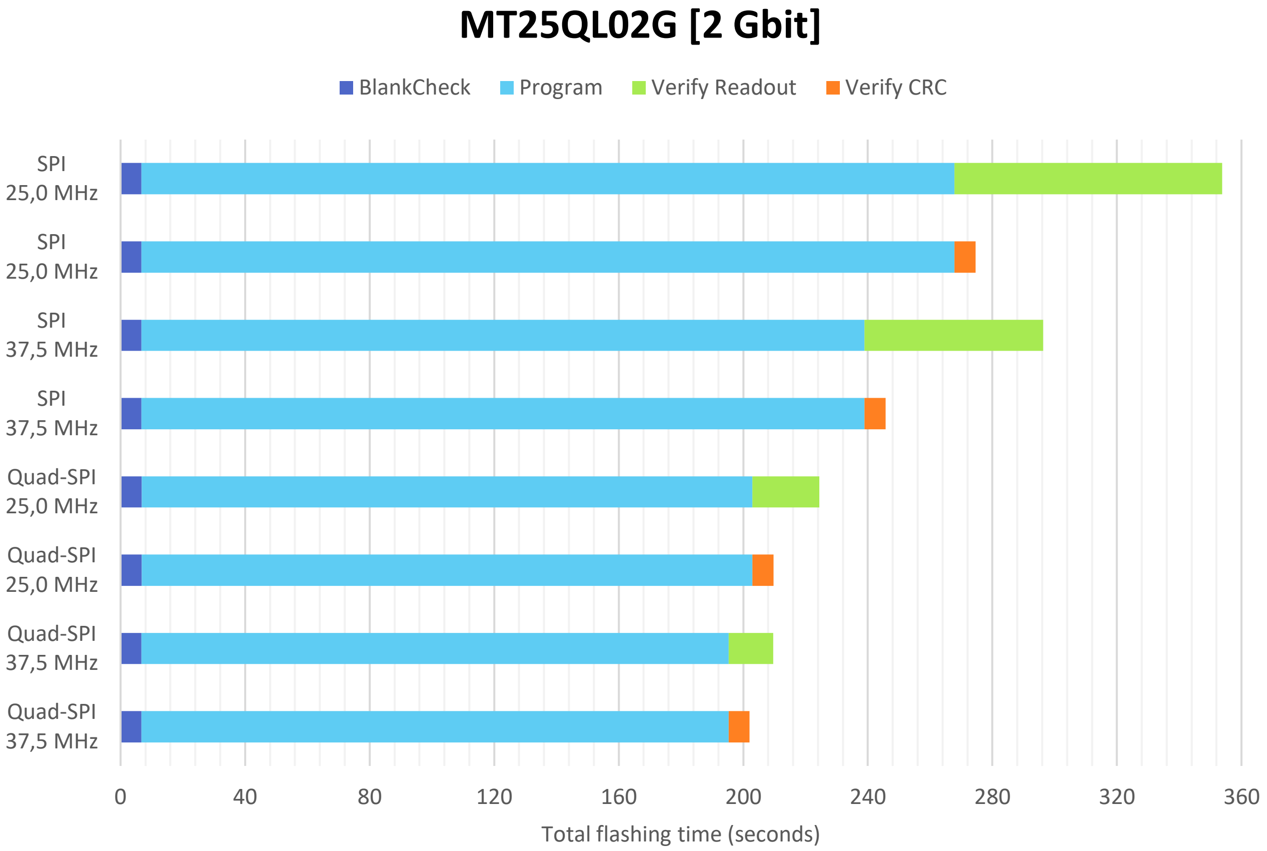
These are some examples of flashing times using our new drivers and OS > 3.00.

Additional notes:

1. Times were measured using FRB files containing random data to cover the entire memory of the target device.
2. We do not consider the masserase time in the total flashing time because it is usually skipped.
3. The time indicated for masserase is related to a virgin device.

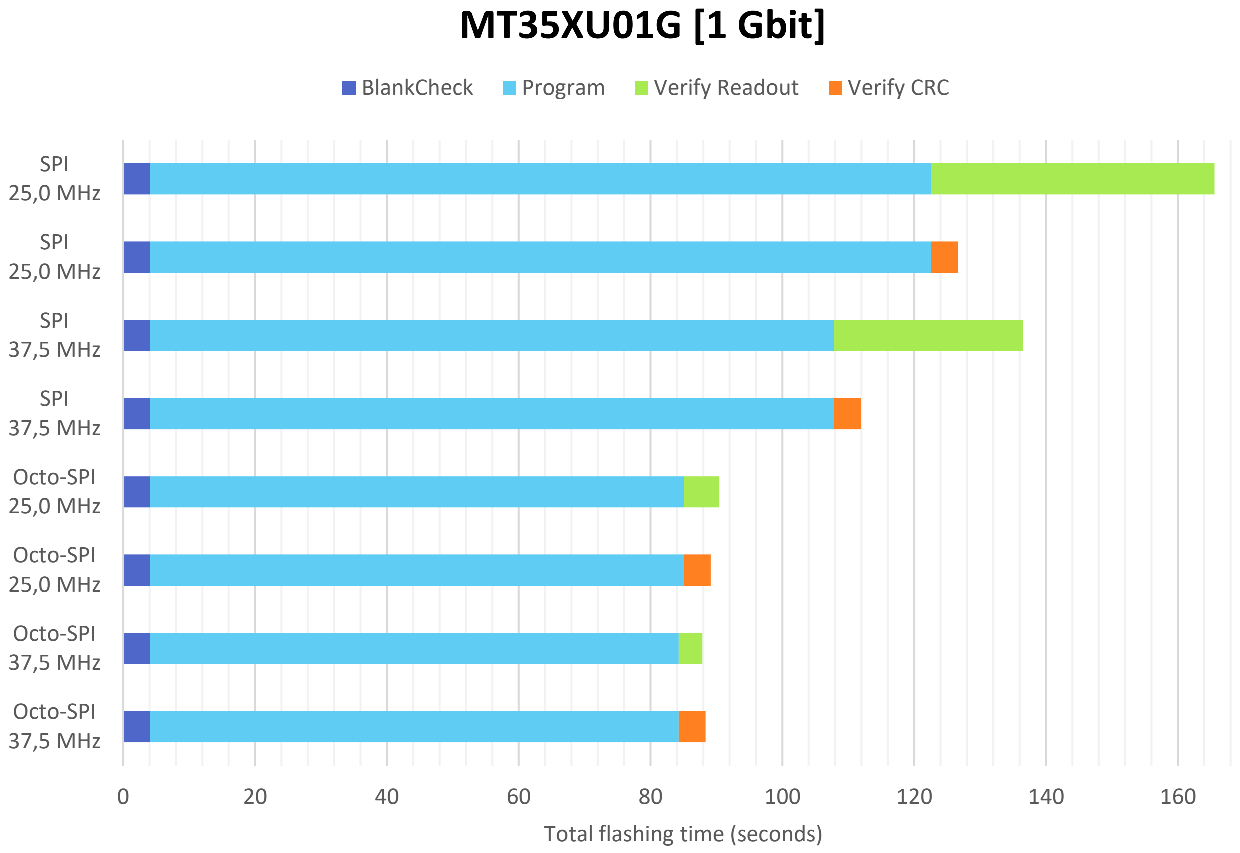


| Protocol | Masserase | BlankCheck | Program | Verify |
|---|---|---|---|---|
| SPI<br>25.0 MHz | 149.91 s | 21.48 s | 154.44 s | 21.49 s |
| SPI<br>37.5 MHz | 149.90 s | 14.32 s | 146.89 s | 14.33 s |
| Quad-SPI<br>25.0 MHz | 149.90 s | 5.37 s | 138.59 s | 5.39 s |
| Quad-SPI<br>37.5 MHz | 149.89 s | 3.58 s | 136.79 s | 3.59 s |



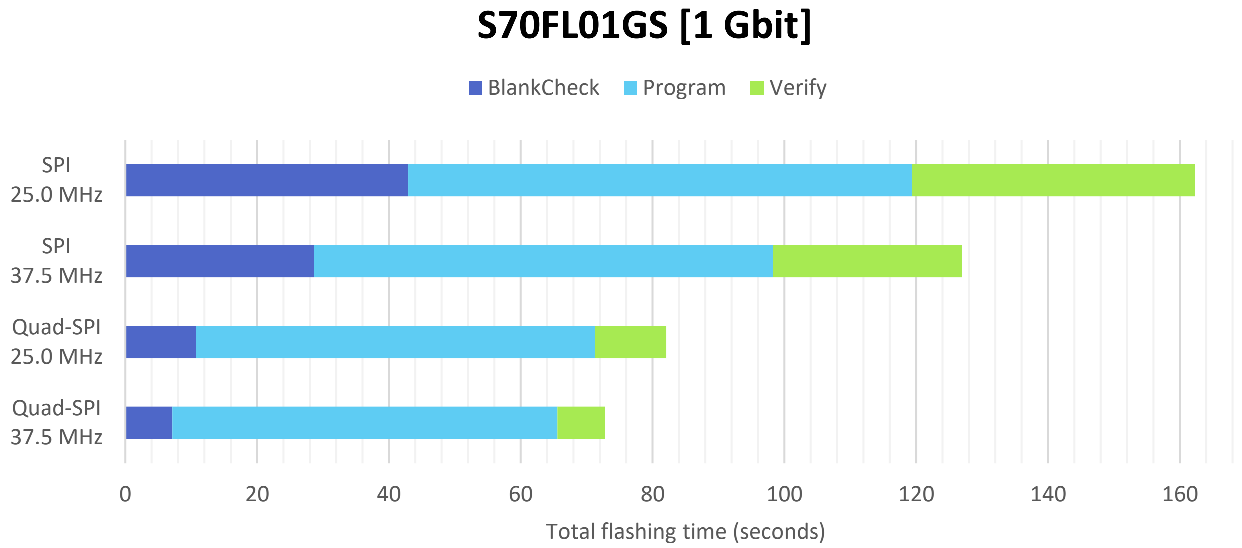
| Protocol | Masserase | BlankCheck | Program | Verify | Verify CRC |
|---|---|---|---|---|---|
| SPI<br>25.0 MHz | 3.35 s | 6.72 s | 261.14 s | 85.95 s | 6.75 s |
| SPI<br>37.5 MHz | 3.32 s | 6.74 s | 232.23 s | 57.31 s | 6.77 s |
| Quad-SPI<br>25.0 MHz | 3.33 s | 6.76 s | 196.13 s | 21.53 s | 6.79 s |
| Quad-SPI<br>37.5 MHz | 3.31 s | 6.73 s | 188.54 s | 14.36 s | 6.77 s |

Note: blankcheck operation done using CRC.



| Protocol | Masserase | BlankCheck | Program | Verify | Verify CRC |
|---|---|---|---|---|---|
| SPI<br>25.0 MHz | 1.15 s | 4.08 s | 118.51 s | 42.97 s | 4.08 s |
| SPI<br>37.5 MHz | 1.13 s | 4.08 s | 103.75 s | 28.66 s | 4.08 s |
| Octo-SPI<br>25.0 MHz | 1.14 s | 4.07 s | 80.97 s | 5.41 s | 4.09 s |
| Octo-SPI<br>37.5 MHz | 1.14 s | 4.08 s | 80.19 s | 3.61 s | 4.09 s |

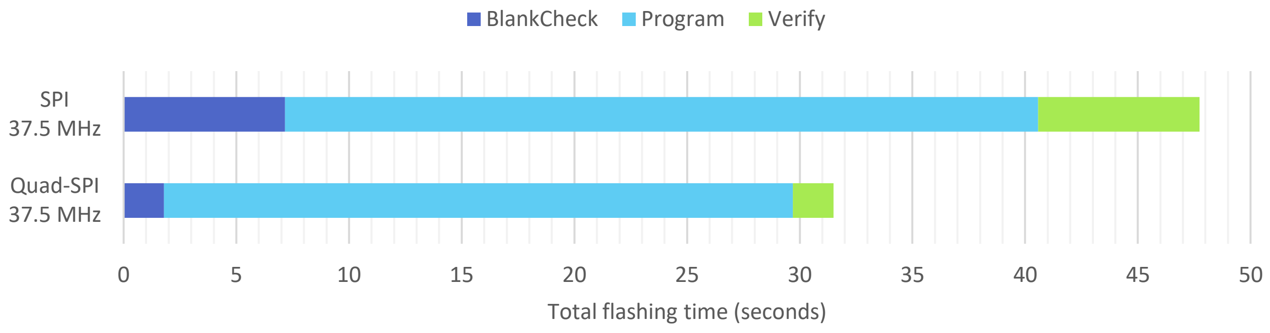
Note: blankcheck operation uses CRC.



| Protocol | Masserase | BlankCheck | Program | Verify |
|---|---|---|---|---|
| SPI<br>25.0 MHz | 114.83 s | 42.95 s | 76.40 s | 42.97 s |
| SPI<br>37.5 MHz | 112.23 s | 28.63 s | 69.65 s | 28.66 s |
| Quad-SPI<br>25.0 MHz | 111.06 s | 10.74 s | 60.54 s | 10.78 s |
| Quad-SPI<br>37.5 MHz | 114.40 s | 7.16 s | 58.42 s | 7.19 s |

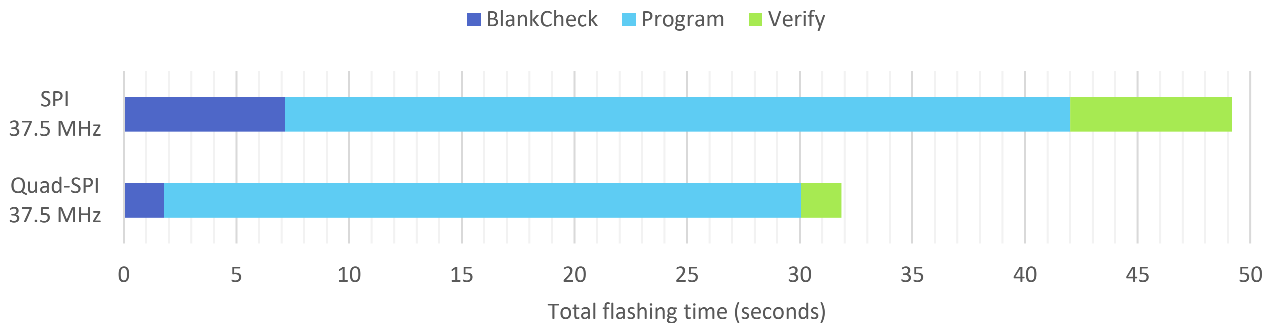
Note: S70FL01GS is a device containing two S25FL512S sharing the same bus and they are managed as described in the chapter "Multiple memories on the same bus".

## IS25LP256D [256 Mbit]

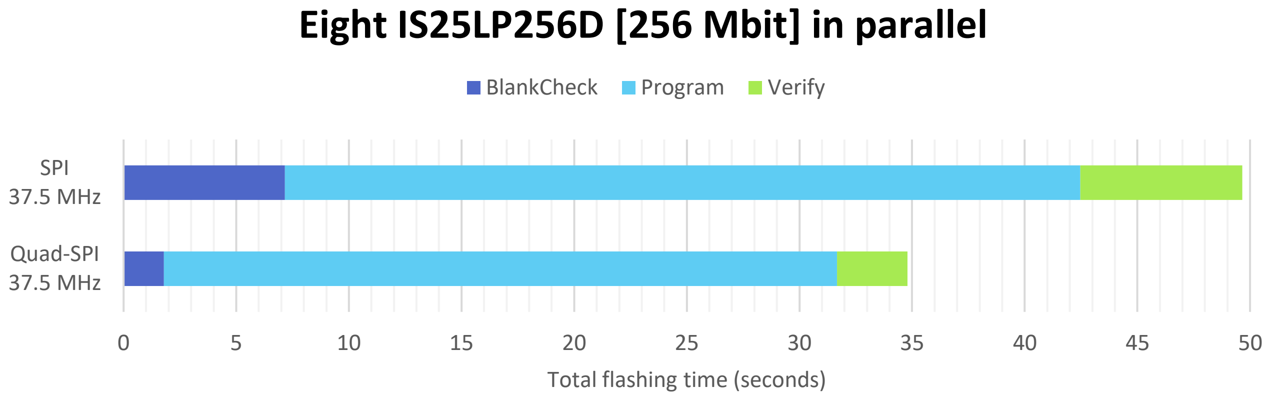


| Protocol | Masserase | BlankCheck | Program | Verify |
|---|---|---|---|---|
| SPI<br>37.5 MHz | 63.81 s | 7.16 s | 33.42 s | 7.16 s |
| Quad-SPI<br>37.5 MHz | 62.46 s | 1.79 s | 27.91 s | 1.80 s |

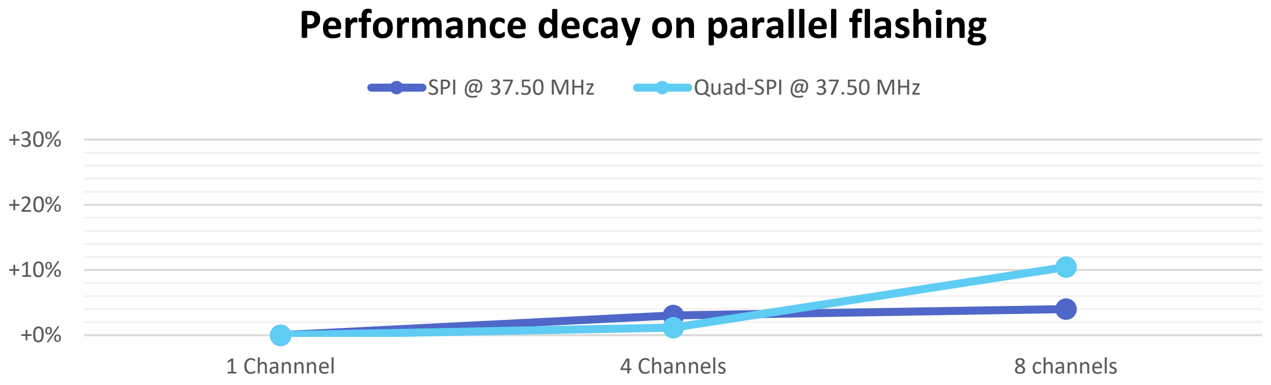
## Four IS25LP256D [256 Mbit] in parallel



| Protocol | Masserase | BlankCheck | Program | Verify |
|---|---|---|---|---|
| SPI<br>37.5 MHz | 63.81 s | 7.16 s | 34.85 s | 7.17 s |
| Quad-SPI<br>37.5 MHz | 62.46 s | 1.79 s | 28.27 s | 1.80 s |


Eight IS25LP256D [256 Mbit] in parallel

| Protocol | Masserase | BlankCheck | Program | Verify |
|---|---|---|---|---|
| SPI<br>37.5 MHz | 63.81 s | 7.16 s | 35.31 s | 7.18 s |
| Quad-SPI<br>37.5 MHz | 62.46 s | 1.79 s | 29.88 s | 3.12 s |


Performance decay on parallel flashing

From the graph above, it is possible to notice that there is a very small decay of the performances on parallel flashing, but it is actually irrelevant for most applications.

This is decay is proportional to the bitrate chosen and the number of parallel channels: it is justified by the fact that FlashRunner has to manage a very big amount of data in a very short time. For example, having eight channels working in Quad-SPI at 37.5MHz, it would mean to ideally manage 1.2 Gbps, which is an extremely huge value.

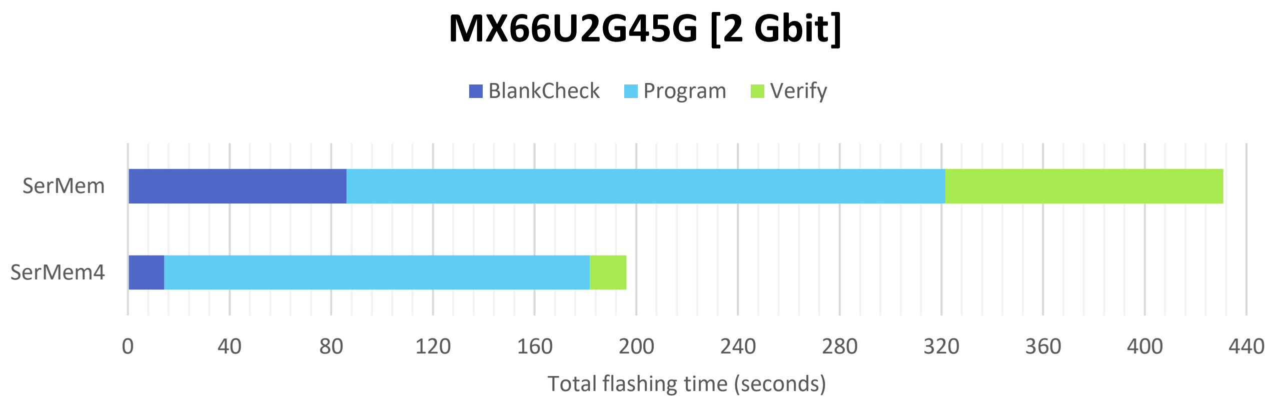
It may seem strange seeing Quad-SPI having less decay than SPI when using four parallel channels, this happens because less time is needed to send data during the program operation, so the FlashRunner can send more data while waiting for other devices to complete the operation. Watching this effect from another point of view, we can say that this kind of decay also depends on device timings, in fact, if the program page operation is slow, then the FlashRunner has more time to do other operations in parallel.

## 10. Performance improvements

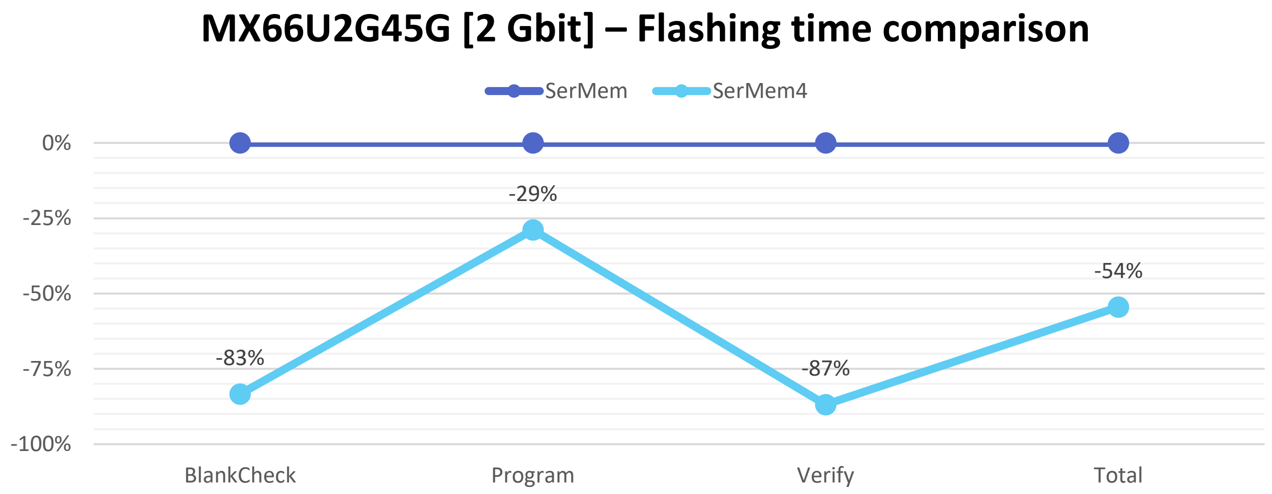
In this chapter we are comparing the top performances of the old SerMem driver with the ones of the new SerMem4 driver to evaluate the improvements.

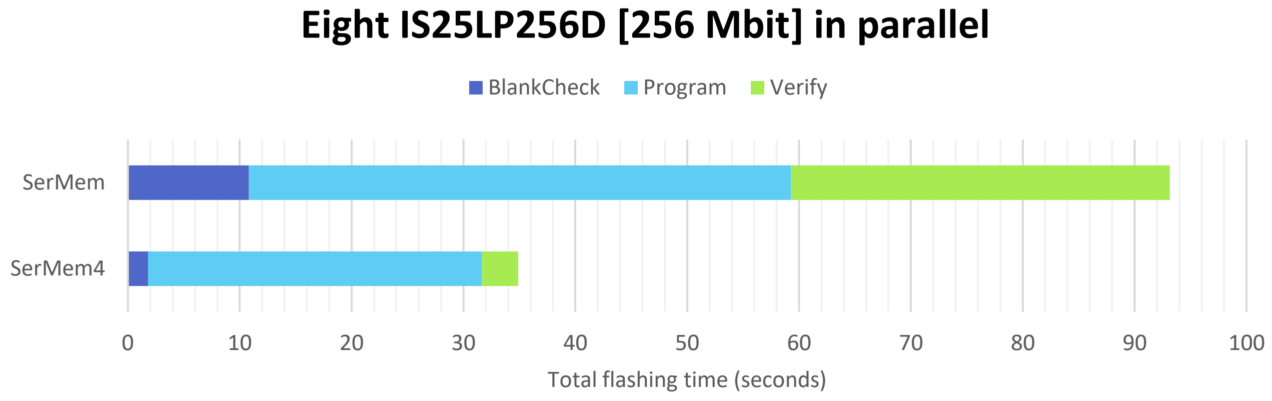
Test conditions:

| Driver name | Driver version | OS version | Protocol | Frequency |
|---|---|---|---|---|
| SerMem | 4.04 | 2.31 | SPI | 25.0 MHz |
| SerMem4 | 1.00 | 3.05 | Quad-SPI | 37.5 MHz |

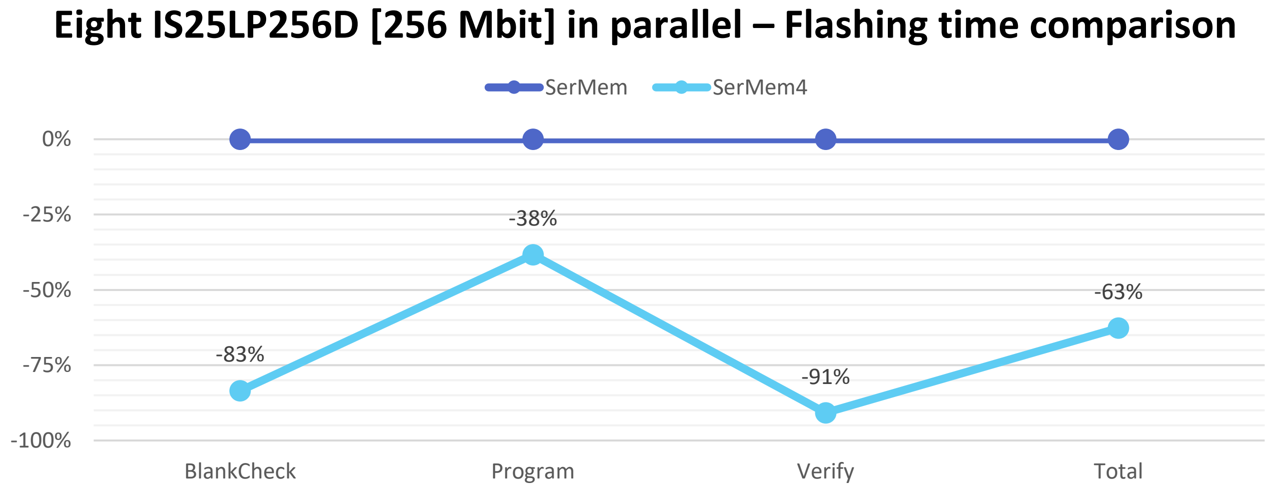


| Driver | Masserase | BlankCheck | Program | Verify |
|---|---|---|---|---|
| SerMem | 188.41 s | 86.08 s | 235.46 s | 109.36 s |
| SerMem4 | 188.33 s | 14.32 s | 167.46 6 | 14.37 s |

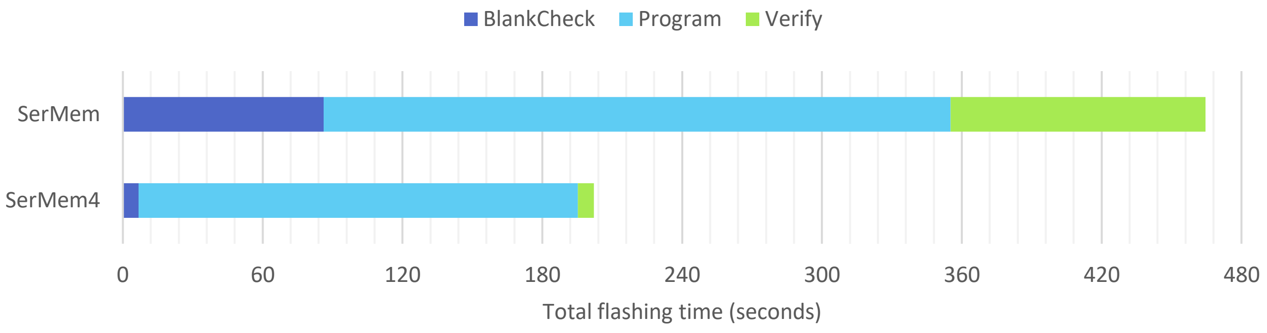




| Driver | Masserase | BlankCheck | Program | Verify |
|---|---|---|---|---|
| SerMem | 64.38 s | 10.83 s | 48.46 s | 33.86 s |
| SerMem4 | 62.46 s | 1.79 s | 29.88 s | 3.12 s |

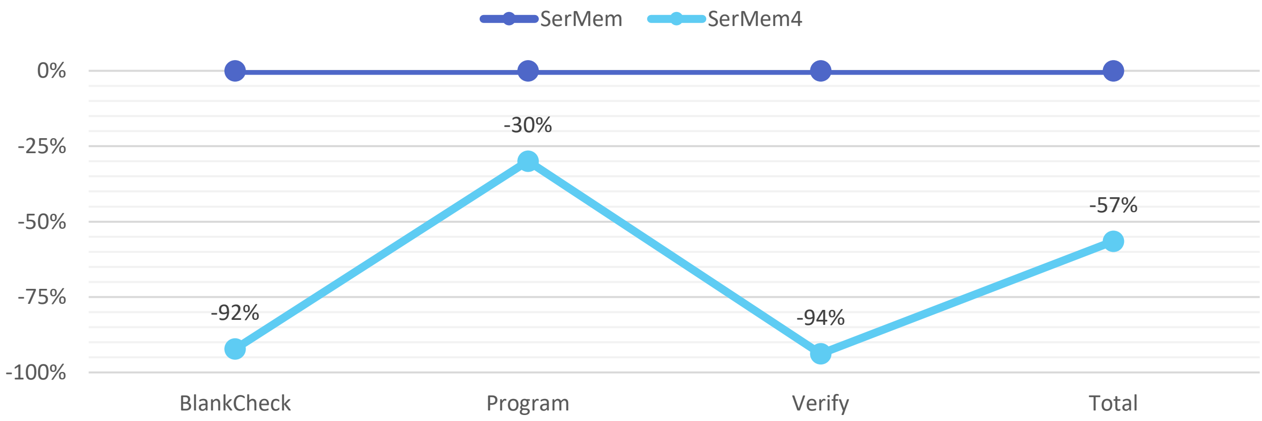


## MT25QL02G [2 Gbit]

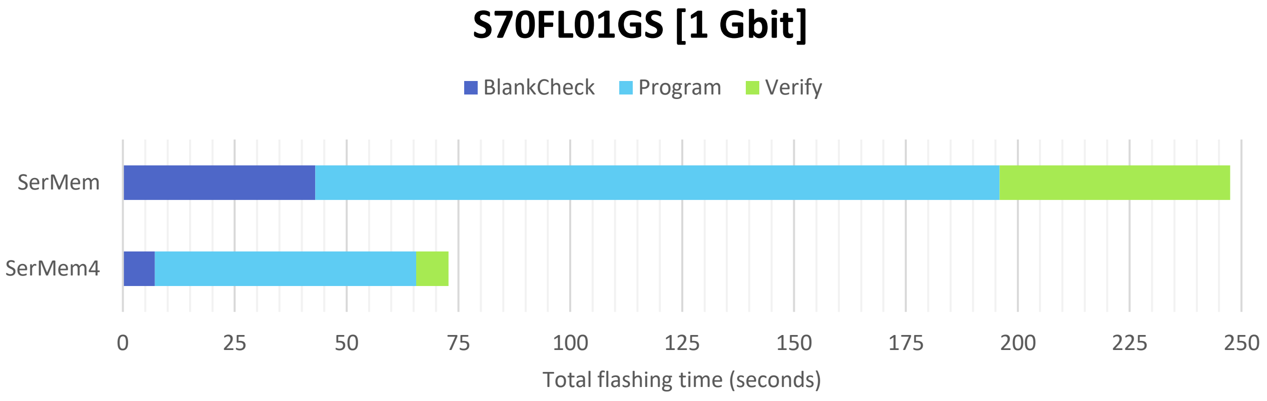


| Driver | Masserase | BlankCheck | Program | Verify |
|---|---|---|---|---|
| SerMem | 3.42 s | 86.08 s | 269.12 s | 109.36 s |
| SerMem4 | 3.31 s | 6.73 s | 188.58 s | 6.77 s |

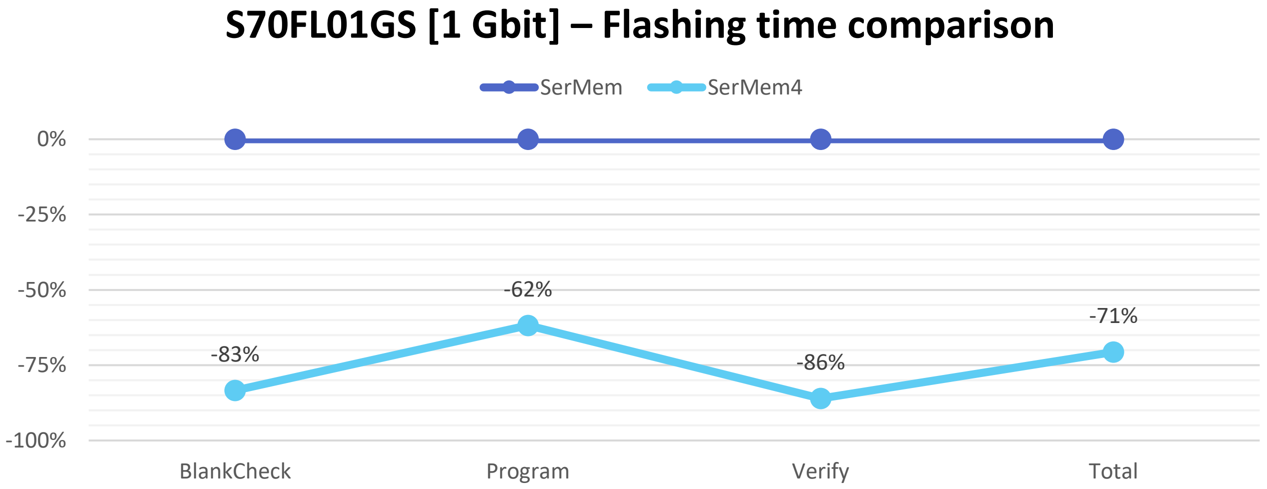
## MT25QL02G [2 Gbit] – Flashing time comparison



Note: for SerMem4, blankcheck and verify operations use CRC.



| Driver | Masserase | BlankCheck | Program | Verify |
|---|---|---|---|---|
| SerMem | 114.93 s | 43.04 s | 152.92 s | 51.55 s |
| SerMem4 | 114.40 s | 7.16 s | 58.42 s | 7.19 s |



Note: S70FL01GS is a device containing two S25FL512S sharing the same bus and they are managed as described in the chapter “Multiple memories on the same bus”.

## 11. Devices which changed the part number

Four devices, which were previously supported in the old SerMem driver, have been renamed in the new driver SerMem4. This has been done to unify them in a more generic part number and to avoid any doubts about compatibility with equivalent devices.

These are the devices which changed the part number:

| Old – SerMem | New – SerMem4 |
|---|---|
| IS25LP128J | IS25LP128 |
| S25FS512SDS | S25FS512S |
| S25FL128S_512 | S25FL128S |
| S25FL256S_512 | S25FL256S |

## 12. Devices which can be upgraded

This is the list of the upgradable devices because they were moved from SerMem to SerMem4:

- **Cypress (Spansion)**

| | | | |
|---|---|---|---|
| S25FL127S | S25FL128S | S25FL256S | S25FL512S |
| S25FS512S | S70FL01GS | | |

- **ISSI**

| | | | |
|---|---|---|---|
| IS25LQ512B | IS25WP020D | IS25LQ040B | IS25LP016D |
| IS25LP032D | IS25LP128 | IS25WP256D | IS25LP256D |
| IS25LP512M | | | |

- **Macronix**

| | | | |
|---|---|---|---|
| MX25V8035F | MX25V1635F | MX25R3235F | MX25L3233F |
| MX25L6433F | MX25L12833F | MX25L12835F | MX25L12845G |
| MX25L25645G | MX66L25685G | MX66L51285G | MX25L51245G |
| MX66L1G45G | | | |

- **Micron**

| | | | |
|---|---|---|---|
| MT25QL128 | MT25QL256 | MT25QL512 | MT25QU01G |
| MT25QL01G | MT25QL02G | | |

- **Winbond**

| | | | |
|---|---|---|---|
| W25Q40CV | W25Q80BL | W25Q80BV | W25Q80BW |
| W25Q80DV | W25Q16BV | W25Q16CV | W25Q16DW |
| W25Q16FW | W25Q16JV | W25Q16V | W25Q32BV |
| W25Q32DW | W25Q32FV | W25Q64CV | W25Q64FV |
| W25Q64JV | W25Q128BV | | |

## 13. Upgrade procedure

In case you are currently using devices mentioned in the chapters "Devices which changed the part number" or "Devices which can be upgraded" and you want to upgrade your system, first you need to contact our support team (support@smh-tech.com) to get the updated licenses, specifying the part number of your devices and the serial number of your FlashRunner.

Meanwhile licenses are arriving, you can follow this procedure to update all the rest. Please, before proceeding, backup your current projects and take note of the versions of the FlashRunner OS and of the SerMem driver you are currently using (just in case something goes wrong). We recommend using the latest versions of the FlashRunner OS, of the Workbench, and of the SermMem4 driver:

- Download FlashRunner OS
- Download Workbench
- Download SerMem4

From the Workbench, press the orange button on the top-right to update your local copy of our database and then create a new project for your device. Using the project editor tab from the Workbench (or using a generic text editor), compare your new project with the one you backed up before. Check that the following fields are present and equal, otherwise, just copy them from the backup project into the new one:

- **`!ENGINEMASK`** *(This value can be expressed both in decimal or hexadecimal)*
- **`#TCSETPAR PROTCLK`**
- **`#TCSETPAR PWDOWN`**
- **`#TCSETPAR PWUP`**
- **`#TCSETPAR VPROG0`**
- **`#TCSETPAR VPROG1`** *(optional)*
- **`#TCSETPAR WD_FREQUENCY`** *(optional)*
- **`#TCSETPAR WD_DUTYCYCLE`** *(optional)*
- **`#TCSETPAR CMODE`** *(In case you were using "QSPI", then use "QUAD-SPI")*
- **`#TPSETSRC`** *(optional)*

At this point, everything before the **`#TPSTART`** command should be ready. So, you can copy-paste everything after the **`#TPSTART`** command from the backup version to the new project because that part must be equal.

Send the new project to the FlashRunner and try to launch its execution. If everything works well, then you can rename the new project with the same as your old project, so you do not need to edit anything else of your application.

See also the example on the next page and, if you have doubts, please contact us to get helped.

This is an example of a project upgrade from SerMem to SerMem4 for a device that also changed the part number from S25FS512SDS to S25FS512S.

| Old project for S25FS512SDS | Upgraded project for S25FS512S |
|---|---|
| !ENGINEMASK 385 | !ENGINEMASK 0x00000181 |
| #LOADDRIVER libsermem.so SPANSION 25 S25FS512SDS | #LOADDRIVER libsermem4.so CYPRESS S25FS_S S25FS512S |
| #TCSETDEV VDDMIN 1700<br>#TCSETDEV VDDMAX 2000<br>...<br>!CRC 2285420621 | #TCSETDEV VDDMIN 1700<br>#TCSETDEV VDDMAX 2000<br>...<br>!CRC 0x10DF8569 |
| #TCSETPAR DEVICE_ADDR 0<br>#TCSETPAR PAGEMODE 0<br>#TCSETPAR PROTCLK 12500000<br>#TCSETPAR PWDOWN 100<br>#TCSETPAR PWUP 100<br>#TCSETPAR RSTDOWN 100<br>#TCSETPAR RSTDRV OPENDRAIN<br>#TCSETPAR RSTUP 100<br>#TCSETPAR S25FL032P_MANDATORYPROCEDURE NO<br>#TCSETPAR VPROG0 1800<br>#TCSETPAR VPROG1 12000<br>#TCSETPAR CMODE QSPI<br>#TCSETPAR WD_FREQUENCY 80<br>#TCSETPAR WD_DUTYCYCLE 50<br>#TPSETSRC 64MB.frb IGNORE_BLANK_PAGE | #TCSETPAR PROTCLK 12500000<br>#TCSETPAR PWDOWN 100<br>#TCSETPAR PWUP 100<br>#TCSETPAR RSTDOWN 100<br>#TCSETPAR RSTDRV OPENDRAIN<br>#TCSETPAR RSTUP 100<br>#TCSETPAR VPROG0 1800<br>#TCSETPAR VPROG1 12000<br>#TCSETPAR CMODE QUAD-SPI<br>#TCSETPAR WD_FREQUENCY 80<br>#TCSETPAR WD_DUTYCYCLE 50<br>#TPSETSRC 64MB.frb IGNORE_BLANK_PAGE |
| #TPSTART<br>#TPCMD CONNECT<br>#IFERR TPCMD BLANKCHECK F<br>#THEN TPCMD MASSERASE F<br>#THEN TPCMD BLANKCHECK F<br>#TPCMD PROGRAM F<br>#TPCMD VERIFY F R<br>#TPCMD PROGRAM S 0x00<br>#TPCMD VERIFY S 0x00<br>#TPCMD PROGRAM R 0x02 0x02<br>#TPCMD VERIFY R 0x02 0x02<br>#TPCMD DISCONNECT<br>#TPEND | #TPSTART<br>#TPCMD CONNECT<br>#IFERR TPCMD BLANKCHECK F<br>#THEN TPCMD MASSERASE F<br>#THEN TPCMD BLANKCHECK F<br>#TPCMD PROGRAM F<br>#TPCMD VERIFY F R<br>#TPCMD PROGRAM S 0x00<br>#TPCMD VERIFY S 0x00<br>#TPCMD PROGRAM R 0x02 0x02<br>#TPCMD VERIFY R 0x02 0x02<br>#TPCMD DISCONNECT<br>#TPEND |

Note: we maintained backward compatibility for register operations even if we have not mentioned that in the chapter "Register operations". This means that you can still use the **PROGRAM** and the **VERIFY** commands with **S**, **C**, or **R** as parameters.

In case you do not feel comfortable doing these operations by yourself, feel free to ask our support team to update your projects for you.

## 14. Devices supported

This is the list of all the currently supported devices:

- **Adesto [Quad-SPI]**

| | | | |
|---|---|---|---|
| AT25QL321 | AT25QF641 | AT25QF641B | AT25QL641 |
| AT25QF128A | AT25QL128A | | |

- **Cypress [Quad-SPI]**

| | | | |
|---|---|---|---|
| S25FL064L | S25FL128L | S25FL256L | S25FL127S |
| S25FL128S | S25FL256S | S25FL512S | S25FS064S |
| S25FS128S | S25FS256S | S25FS512S | S70FL01GS |

- **ISSI [Quad-SPI]**

| | | | |
|---|---|---|---|
| IS25LQ025B | IS25LQ512B | IS25LQ010B | IS25LQ020B |
| IS25WQ020 | IS25WP020D | IS25LQ040B | IS25WQ040 |
| IS25WP040D | IS25LP080D | IS25WP080D | IS25LP016D |
| IS25WP016D | IS25LP032D | IS25WP032D | IS25LP064A |
| IS25LP064D | IS25WP064D | IS25LP128 | IS25LP128F |
| IS25WP128 | IS25WP128F | IS25LP256D | IS25LP256E |
| IS25WP256D | IS25WP256E | IS25LP512M | IS25LP512MG |
| IS25WP512M | IS25WP512MG | IS25LP01G | IS25WP01G |

- **Macronix [Quad-SPI]**

| | | | |
|---|---|---|---|
| MX25R512F | MX25R1035F | MX25R2035F | MX25R4035F |
| MX25R8035F | MX25R1635F | MX25R3235F | MX25R6435F |
| MX25V2033F | MX25V2039F | MX25V4035F | MX25V8035F |
| MX25V1635F | MX25U1001E | MX25U2032E | MX25U2033E |
| MX25U2035F | MX25U4032E | MX25U4035F | MX25U8033E |
| MX25U8035F | MX25U1633F | MX25U3232F | MX25U6432F |
| MX25U6472F | MX25U12832F | MX25U12843G | MX25U12845G |
| MX25U12872F | MX25U25645G | MX25U25673G | MX25U5121E |
| MX25U51245G | MX66U1G45G | MX66U2G45G | MX25L8036E |
| MX25L1636E | MX25L1655D | MX25L1673E | MX25L3233F |
| MX25L3236F | MX25L3255E | MX25L3273F | MX25L6433F |
| MX25L6435E | MX25L6436F | MX25L6456E | MX25L6456F |
| MX25L6473F | MX25L12833F | MX25L12835F | MX25L12845G |
| MX25L12850F | MX25L12855F | MX25L12865F | MX25L12872F |

| | | | |
|---|---|---|---|
| MX25L12873G | MX25L25633F | MX25L25635E | MX25L25635F |
| MX25L25645G | MX25L25672F | MX25L25673G | MX25L25733F |
| MX25L25745G | MX25L51237G | MX25L51245G | MX25L51255G |
| MX25L51273G | MX66L1G45G | MX66L1G55G | MX66L1G59G |
| MX66L2G45G | MX66L25685G | MX66L51285G | MX66L1G85G |
| MX25U25645G_54 | MX25U51245G_54 | MX66U1G45G_54 | MX66U2G45G_54 |
| MX77L6450F | MX77L12850F | MX77U25650F | MX77L25650F |

- **Micron [Quad-SPI]**

| | | | |
|---|---|---|---|
| MT25QL128 | MT25QU128 | MT25QL256 | MT25QU256 |
| MT25QL512 | MT25QU512 | MT25QL01G | MT25QU01G |
| MT25QL02G | MT25QU02G | | |

- **Winbond [Quad-SPI]**

| | | | |
|---|---|---|---|
| W25Q10EW | W25Q20CL | W25Q20CV | W25Q20EW |
| W25Q40CL | W25Q40CV | W25Q40EW | W25Q80BL |
| W25Q80BV | W25Q80BW | W25Q80DL | W25Q80DV |
| W25Q80EW | W25Q16BV | W25Q16CL | W25Q16CV |
| W25Q16DV | W25Q16DW | W25Q16FW | W25Q16JL |
| W25Q16JV | W25Q16JV_DTR | W25Q16JV_QE | W25Q16JW |
| W25Q16JW_DTR | W25Q16JW_QE | W25Q16V | W25Q32BV |
| W25Q32DW | W25Q32FV | W25Q32FV_QE | W25Q32FW |
| W25Q32JV | W25Q32JV_DTR | W25Q32JV_QE | W25Q32JW |
| W25Q32JW_DTR | W25Q32JW_QE | W25Q64BV | W25Q64CV |
| W25Q64FV | W25Q64FV_QE | W25Q64FW | W25Q64JV |
| W25Q64JV_DTR | W25Q64JV_QE | W25Q64JW | W25Q64JW_DTR |
| W25Q64JW_QE | W25Q128BV | W25Q128FV | W25Q128FV_QE |
| W25Q128FW | W25Q128JV | W25Q128JV_DTR | W25Q128JV_QE |
| W25Q128JW | W25Q128JW_DTR | W25Q128JW_QE | W25Q256FV |
| W25Q256FV_QE | W25Q256JV | W25Q256JV_DTR | W25Q256JV_QE |
| W25Q256JW | W25Q256JW_DTR | W25Q256JW_QE | W25Q257FV |
| W25Q257FV_QE | W25Q257JV | W25Q512JV | W25Q512JV_DTR |
| W25Q01JV | W25Q01JV_DTR | W25M512JV | W25M512JW |
| W25R64FV | W25R64JV | W25R64JW | W25R128FV |
| W25R128FW | W25R128JV | W25R128JW | W25R256JV |
| W25R256JW | | | |

- **Adesto [Octo-SPI]**

| | | | |
|---|---|---|---|
| ATXP032 | ATXP064 | ATXP064B | ATXP128 |

- **Cypress [Octo-SPI]**

| | | | |
|---|---|---|---|
| S28HL256T | S28HS256T | S28HL512T | S28HS512T |
| S28HL01GT | S28HS01GT | | |

- **ISSI [Octo-SPI]**

| | | | |
|---|---|---|---|
| IS25LX064 | IS25WX064 | IS25LX128 | IS25WX128 |
| IS25LX256 | IS25WX256 | IS25LX512 | IS25WX512 |
| IS25LX01G | IS25WX01G | IS25LX02G | IS25WX02G |

- **Macronix [Octo-SPI]**

| | | | |
|---|---|---|---|
| MX25UW63 | MX25UW64 | MX25UM128 | MX25UW123 |
| MX25UW128 | MX25UW12A | MX25LM256 | MX25UM253 |
| MX25UM256 | MX25UW253 | MX25UW256 | MX25LM512 |
| MX25UM512 | MX25UM512_ExS | MX25UM513 | MX25LW512 |
| MX25UW512 | MX25UW513 | MX66LM1G | MX66UM1G |
| MX66LW1G | MX66UW1G | MX66LM2G | MX66UM2G |
| MX66LW2G | MX66UW2G | | |

- **Micron [Octo-SPI]**

| | | | |
|---|---|---|---|
| MT35XL256 | MT35XU256 | MT35XL512 | MT35XU512 |
| MT35XL01G | MT35XU01G | MT35XL02G | MT35XU02G |
| MT35XU512_x3 | | | |